Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/5510--98-8146

NAC: An Adaptive Case-Based Reasoning Tool for Experimenting with Retrieval and Indexing

LiWu Chang
Patrick R. Harrison
Laura C. Davis

Navy Center for Applied Research in Artificial Intelligence
Information Technology Division

March 13, 1998

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 13, 1998	

4. TITLE AND SUBTITLE

NAC: An Adaptive Case-Based Reasoning Tool for Experimenting with Retrieval and Indexing

5. FUNDING NUMBERS

PN - 55-6470
PE - 62234N
TA - IT4101

6. AUTHOR(S)

LiWu Chang, Patrick R. Harrison, and Laura C. Davis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5510--98-8146

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

NAC is a testbed for experimenting with concepts of retrieval and indexing in Case-Based Reasoning (CBR). The paper describes similarity functions and decision functions used for retrieval as well as methods for re-indexing and case organization. The paper also describes methods for weighting attributes, analyzing their dependence and evaluating the importance index of a single stored case. Two examples of how to use NAC are provided. Methods employed for retrieval and indexing are based on mathematically sound techniques developed in classification, clustering and decision analysis. NAC includes basic functions for specifying similarity, normalizing data and evaluation. Retrieval is done using both nonparametric (e.g., nearest neighbor) and parametric (i.e., Bayesian) statistical procedures with weighted attributes. New indices for cases are generated using clustering methods. Cases are re-organizing using the new indices. NAC also allows the user to test the predictive accuracy of retrieval methods and the quality of indices generated in (re)-indexing. NAC includes adaptive functions for enhancing the performance of retrieval and indexing. Important functions for adaptation include weighting and selecting attributes, learning dependency relationships and calculating typicality for each stored case. The NAC environment was designed so that additional techniques and metrics can easily be added.

14. SUBJECT TERMS

Case-based reasoning
Retrieval
Indexing
Similarity
Clustering
Nearest Neighbor

15. NUMBER OF PAGES

21

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

NAC: An Adaptive CBR Tool for Experimenting with Retrieval and Indexing

1 Introduction

The goal of Case-Based Reasoning (CBR) is to effectively use past experiences in problem solving. CBR has at its core, *adaptive* methods for memory indexing, retrieval and organization. NAC was designed as a testbed for studying these adaptive memory functions for retrieval, indexing and organization on real-world problems. The evolution of NAC was based on an abstract model of memory that provides a theorical basis for NAC analogs of human capacities to classify, cluster, organize, index and retrieve information. [Harrison, Chang & Meyrowitz, 1994].

Adaptation is generally viewed as a process for achieving optimum results. Contents of adaptive methods range from the control of reflexive responses to knowledge compilation and concept formation. We consider two aspects of adaptation which are relevant to retrieval and indexing functions in memory based reasoning. The first aspect concerns dynamic updating of similarity measurements for case retrieval. In NAC, the computation of similarity is like evidence combination which pools the measure of difference of each attribute. Factors, such as the importance weights and dependency relationships of attributes, that have great impact on the computation, are accounted for as the components of adaptation. In dynamic environments, the importance weights of attributes vary as new events occur. Consequently, the similarity measurements are also updated.

The second aspect concerns the organization of cases. One reason for organizing cases is to facilitate coherent retrieval and control search attention. In NAC, new indices are generated using clustering methods (e.g., incremental clustering). Adaptive procedures are used to update clusters in incremental clustering. Context or perspective information may be incorporated in case organization as latent factors, each corresponding to a subset of attributes. As a result, different cases can be retrieved from different perspectives.

Figure 1 shows the top level NAC interface. It provides a user interface designed for experimentation with the analogs mentioned above. It provides support for running multiple experiments and comparing results. Section 6 describes two examples of the use of NAC, and Appendices 1 and 2 provide additional information for the NAC software user. NAC provides the means to implement models of memory and to do case-based reasoning. It is a testbed for testing concepts in retrieval and (re)-indexing. A variety of functions have been implemented in NAC. These include similarity metrics, adaptive functions and methods of classification and clustering for organizing cases. NAC is intended to be a scalable tool. It was designed so that any number of new similarity, normalization and evaluation functions can be added with a minimum of effort. The same idea applies to the adaptive and classification functions. The discussion here does not imply that the functions provided in NAC are inclusive. They should be considered a typical starting set.

Manuscript approved February 2, 1998

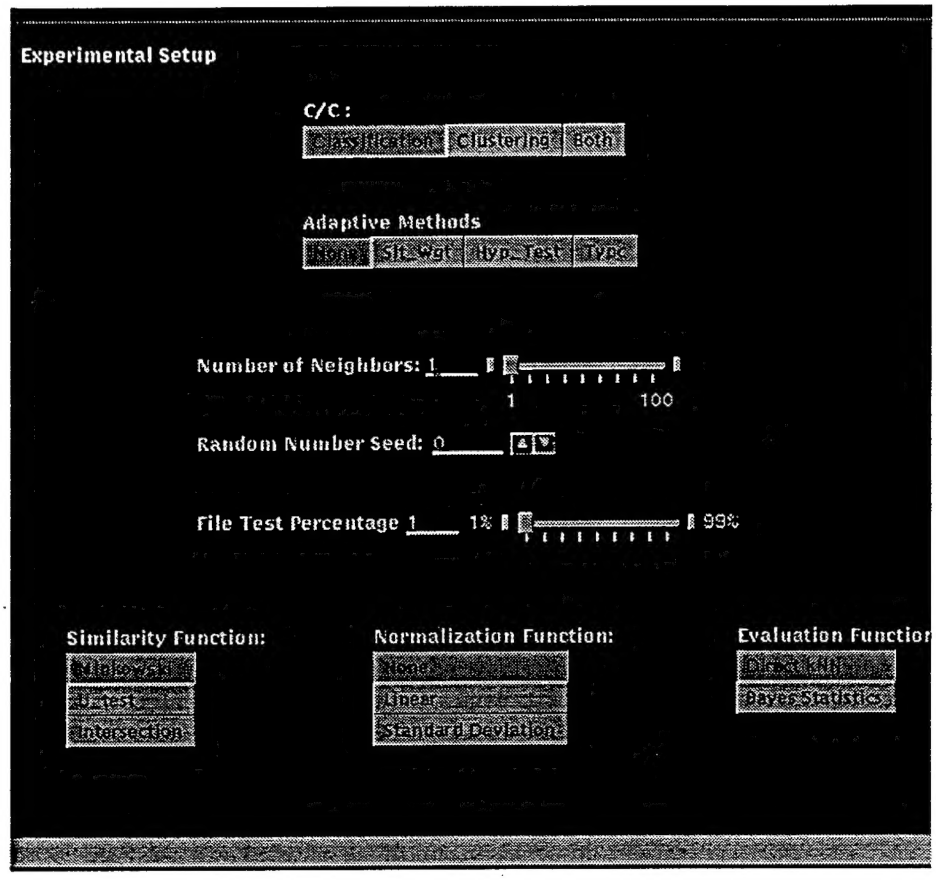

Figure 1. NAC Top-Level Interface

2 Basic Structures

A case may have structures of great complexity. Nonetheless, the reference to a case often can be summarized by the index set. A case such as one concerning the Gulf war could be encoded with indices of humanity, oil supply, territorial dispute and so on. Under this assumption, our discussion of case retrieval and organization is in the light of the computation of the index set. For routine retrieval, cases may be extensively indexed to speed search. For novel retrieval, index sets are likely to be revised, transformed and expanded over time. There may exist structural relationship among indices. One objective of case organization is to explore the hidden structures.

The basic type of index representation used in NAC is the attribute list which is probably the dominant type in many applications (e.g., database). The value of attributes may be of nominal, ordinal or interval scale. Inference and decision are carried out with a probability distribution function (referred to as PM, i.e., probability distribution metric) or without one (referred to as DM, i.e., distance metric). In the case of decision without a model (DM), the non-parametric method, voting k-Nearest Neighbor (kNN), procedure is used. On the other hand, in the case of decision with a probabilistic metric (PM), Bayes decision rule is used for estimation when data is of nominal type, while for data of ordinal or inter-

val scale, Parzen approach [Parzen, 1962] with a normal kernel function is used for estimation. For continuous variables, Gaussian distribution is perhaps the most commonly used density function. However, we do not use Gaussian distribution in the current version of NAC due to the fact that many distributions may not fit the Gaussian model well.

3 Case Retrieval

Retrieval is an interactive decision-making process with the goal of selecting cases and ranking the selections. NAC considers three forms of retrieval:

1) Given a target case, retrieve the most similar case. In this situation, the index set of the target is compared against that of each stored case.

2) Given an query, retrieve a case that explains the query best. The query is encoded in terms of the index set and the decision is based on the encoded relations.

3) Retrieve exemplar cases from a category according to information about that category. Decision functions such as Bayesian theory and kNN are employed to handle different forms of retrieval, knowing that they can be used interchangeably in many circumstances. Ranking relies on the given similarity metrics.

3.1 Similarity metrics

The measured distance between data points is the basis for defining similarity. The distance is measured with respect to one attribute. Similarity is obtained from pooling measures over several attributes.

The intersection between attribute lists is used as a similarity metric when data is of nominal scale. The difference of an attribute is either 0 or 1. The similarity of two cases is determined according to the number of 1's divided by the total number of attributes. For data of ordinal or interval scale, the Minkowski metric [e.g., Jain & Dubes, 1988] which generalizes Euclidean distance is a common one,

$$\left| \sum_{j=1}^{J} w_{aj} \times |Xrj - Xsj|^{\lambda} \right|^{1/\lambda}$$

where w is the weight of attribute. The final similarity measure may be multiplied by the weight associated with each stored case.

In case of ordinal or parametric estimation, the proposed similarity metric is based on the difference measure between joint conditional probability (jcp) density functions of two categories (or classes), i.e.,

$$\left[\sum_{l}^{L} \left| p(F_i = l | C_a) - p(F_i = l | C_b) \right|^2 \right]$$

where "*Fi*" stands for the i'th attribute, "*l*" stands for the possible values or states of the i'th attribute and "C" is the category or class label. This equation reaches the minimum 0 when two jcp functions are identical and have the maximum value 2. Note that, in the case of ordinal and interval scaled data, p() is the Parzen function.

NAC provides options for similarity measures. Options include *linearization* and *standardization* where the former metric linearizes the data so that all values fall into the unit interval, and the latter one normalizes the data with standard deviation. Normalization removes the difference between scales associated with measures. For example, standardization takes into account the covariance structure of the data and differentiates those points that are close to the centroid from those that are far away. The negative effect is that normalization removes the correlation information among features which may be important in the computation (e.g., linear regression analysis] [Sachs, 1984]).

3.2 Decision functions for case retrieval

NAC incorporates the voting kNN (nearest neighbor) as the decision metric (DM). The voting kNN bases its decision on the frequency count of the majority class label seen in the first k nearest neighbors. Variations to the frequency count rule are possible. For example, membership count may be used in place of frequency count (i.e., each count is replaced by its membership), where membership can be assigned to 1 over the measure of similarity. The peformance of different kNN models are close. The kNN procedure is viewed as a type of linear kernel function.

If the joint conditional probability distribution (jcp) is available between a query and attributes, a Bayesian decision rule can be applied. The probabilistic network model [Pearl, 1988] is employed when Bayesian theory is used for decision making. In some situations, the jcp needs to be learned inductively from stored cases. In NAC, the jcp is estimated from data using the multi-nominal distribution when data is nominal under the assumption of conjugate prior [Berger, 1985]. We compute the jcp for each attribute. The estimated value of each entry of the jcp, with conjugate prior, is determined by the ratio

$$(1 + n_{fc})/(n_f + n_c)$$

where n_{fc} is the number of data of cluster c with feature value f, n_f is the number of feature values, and n_c stands for the number of data in the cluster c [Anderson & Matessa, 1992]. If n_c grows faster than n_f as more data arrive, this expected value will be close to the frequency count. In the absence of any information, jcp is uniformly distributed with $1/n_f$. Note that the range of values of an attribute is assumed to be known. In the absence of this information, an estimate needs to be given to the total number of possible attribute values.

Data may be missing for some training cases. In the case of missing data, the final result may be obtained from averaging over all possible values that the attribute can take. This approach becomes impractical when the number of training cases is large for a large range of attribute values. In NAC, the unknown value of an attribute is viewed as a new category

for nominal values. In the case of ordinal or interval-scaled values, the missing data are ignored.

When training cases do not cover and we do not know all possible values of attributes, it may be the case that an attribute value of input instances does not equal any existing value of that attribute. An equal likelihood ratio is assumed to account for the ignorance.

The performance of kNN based on different membership counts varies for different inputs. Therefore, NAC uses a traditional voting model. The asymptotic error analysis for voting kNN model has long been documented [e.g., Fukunaga, 1990], while it is unavailable for many other variations of kNN. It is calculated under the assumption that the probability of a particular class given the test data is the same as that of its nearest neighbor. The amount of data required for this calculation is sensitive to dimensionality. Due to the effect of dimensionality, the asymptotic error analysis may not be a good performance indicator. In fact, kNN performs well in many empirical studies with relatively small samples. The reason might be that data occupy only lower dimensions due to high degree of correlation and the dimensionality affects every estimated class probability distribution roughly in the same way [Friedman, 1994].

The most similar case will be returned from a retrieval. In terms of a probabilistic network, we may select a case that maximizes the posterior probability of a particular class. We may also select exemplars from a class with respect to input information. The exemplar is the desired case. The difference between the two methods is that Bayes needs to encode new cases to probability distribution functions. In the case of updating probabilistic information with uncertain evidence, the Bayesian method is useful. Decision-based retrieval has also been used in the area of Information/Text Retrieval and Extraction (e.g., [Del Favero & Fung, 1994][Croft & Turtle, 1993]).

4 Methods for Organizing Cases

Casebase (CB) organization is a process that explores underlying relations among attributes and cases. The result may be a hierarchical representation of stored cases. The new structures can be described by using existing attributes or new terms. A CB structure such as la discriminant network may be indexed with existing weighted attributes, while a a CB structure generated using unsupervised learning needs new terms are needed for each cluster. The current focus of case organization in NAC is on using unsupervised structure learning with clustering methods. In this section, we describe methods for deciding the number of clusters (i.e., the indices) generated from clustering.

The general approach to clustering is to agglomerate areas of similar measurements in a feature space. This holds for many situations except under very noisy conditions. Clustering may be carried out in a hierarchical or an incremental mode. Briefly, hierarchical clustering takes all data into account and usually proceeds from bottom-up, while the incremental approach decides, for each case, whether it should be assigned to one of the existing categories or should be given a new label based on a cluster criterion g. The two modes can be used separately or in combination. NAC uses the nearest neighbor principle

in both modes as the criterion for agglomerating two clusters. We experimented with similarity metrics such as density measure and likelihood. The likelihood results work from a probabilistic model whereas density search control takes advantage of kNN for clustering. We considered not only the clustering form but also the quality of the methods (i.e., the ability to handle various distributions).

4.1 Incremental clustering mode

In the case of the probabilistic metric (PM), clustering proceeds by incrementally updating the conditional probability distribution as new instances are processed. The jcp is initially null. As new instances are incorporated, the jcp expands the number of clusters. On the other hand, in DM, the nearest neighbor distance is updated.

Two ways of assigning new labels are no-revision and a revision strategy. In the first, as clustering proceeds, no revision to the contents of clusters is taken. A cluster label is assigned based on posterior probabilities of the candidate clusters. In NAC, the rejection criterion g for PM is related to the maximum value of probability that a cluster *Cj* could have, i.e.,

$$\prod_i \langle max(P\langle F_i | C_j \rangle) \rangle$$

Suppose the new arrival is assigned to cluster *Cj*. Criterion g measures the ratio between the maximum likelihood in the presence of the new data and the maximum possible probability that the cluster *Cj* could have. The idea is that if the arriving data are truly desirable, the result of the maximum likelihood value in the presence of this data is expected to be close to that of the maximum possible probability. In the case of DM, the maximum likelihood rule is replaced by the density measure of each cluster. Many other forms of *g* can be defined [e.g., Jobson, 1991].

Using the revision strategy in incremental clustering, error at an earlier stage is assumed to have an impact on decisions at a later stage. Revision adjusts the contents of clusters as soon as the label of a new case is assigned. Once a case arrives, NAC updates the average nearest neighbor distance, *S*, decides the label for the new arrival and revises (e.g., split, merge etc.) contents of existing clusters according to *S*. Finally, the new arrival is classified to an existing cluster if its first nearest neighbor distance is smaller than *S*. A case is removed from its cluster if it is at least *S* distance away from other points in the presence of the new arrival.

The no-revision strategy was used in COBWEB [Fisher, 1987]. The revision strategy is similar to adaptive k-means where the means and variance of each cluster is adjusted for each new label re-assignment. This strategy involves computation based on global information. The modified version of COBWEB (Gennari etc., 1989) also mentioned the use of split and merging operators. The difference is that revision in NAC is based on global measurements. The second strategy takes longer time but yields less misplaced cases than the first one. In fact, revision is necessary for any kind of incremental process. NAC imposes a strict criterion for assigning new cases to existing clusters in the incremental

clustering phase and compensates for this by merging in the second phase. If every case is assigned a new cluster label then the clustering becomes a hierarchical approach.

4.2 Hierarchical clustering

In NAC, there are two ways to start hierarchical clustering. In one approach, a hierarchical procedure evaluates the clustering criterion for combination with each cluster containing just one case at the beginning. The second way starts with the search according to the measure of denseness of each case (i.e., mode). The denseness around a case is calculated by counting the number of neighbor cases inside its neighborhood with radius *S*. The search starts with cases that have the highest density and ends with cases that have designated boundary density values. Cases selected in each search path constitute an initial cluster. At this level, NAC can either lower the boundary density value to continue the search or continue with the combining operation.

The criterion for mergence is based on similarity metrics. In the case of DM, the criterion is the shortest distance between two clusters. For PM, NAC averages the distribution difference, i.e,

$$[P(F_i/C_i) - P(F_i/C_j)]^2 \times P(F_i/C_i)$$

over clusters other than *i*. In NAC, the proposed clustering criterion for determining the cluster to be merged differs from the normative nearest distance hierarchical clustering in that distance is not the only determinant factor. Actually, the criterion is based on the similarity measure as well as the number of cases in a cluster. This criterion postpones the processing of large clusters so that it yields clottier clusters than does the pure nearest distance approach.

The optimality criterion for determining the optimal number of new clusters differs in non-parametric and parametric approaches. In the probabilistic model, the optimality criterion is determined by the posterior probability of the number of clusters given stored cases, i.e.,

$$Pr(I|S) = \alpha \left\{ \sum_{n} \left[\int_{T} Pr\langle S|T, I, Bn\rangle Pr\langle T|Bn, I\rangle dT \right] Pr(Bn, I) \right\}$$

where ***I, S, T, Bn*** stands for the number of indices (i.e., clusters), sample set, parameters with structure ***Bn*** and the structure ***Bn*** (i.e., the topology of a network), respectively. The result of clustering differs for different ***Bn***s. For a given ***Bn***, the best number of clusters is the one that maximizes the following expression [Cooper & Herskovits, 1991] with uniform prior of ***I***:

$$(n_f - 1)!/(n_f + n_c - 1)! \times \prod n_{fc}!$$

This expression will be evaluated over all attributes. In DM, it is meaningful to talk about the distance between data points. The optimality criterion is determined as a function of the size of clusters, the intra-cluster density measure and inter-cluster distance measure with respect to cluster *Ci*, i.e.,

$$size(C_i) \times (-\log((IntraClusterDensity(C_i))/(InterClusterDist(C_i))))$$

where the separability measure is defined in terms of inter-cluster distance. The measure of inter-cluster density can be estimated by dispersion analysis. In the current version of NAC, this measure is related to the distance calculated from the minimum spanning tree. A similar form is defined for Parzen functions in PM.

Different I's are ranked according to their *Pr(I|S)* values. To simplify the generated index structure, those I's with high *Pr(I|S)* values are retained. In our experiments with several test data sets, the optimality criterion gives a reasonable prediction as we compare the results with raw data. However, in a fairly randomized pattern, the optimality metric become less informative. In multivariate statistical analysis, the optimality criterion is determined by more than one metric. For noisy data, we experimented with methods for removing ambiguous samples, where ambiguous samples are those samples that are equally likely to be assigned to more than one cluster. We investigated a similar form of the optimality criterion in a probabilistic model for non-parametric functions.

As compared to other clustering systems, such as AUTOCLASS and Anderson's categorizing system, NAC aims at an integrated approach. In NAC, several different clustering approaches were incorporated for comparative analysis.

5 Methods for Calculating Importance Index

NAC considers three types of factors that affect the measure of similarity. The first type of factor is related to selectively weighting attributes in the environment so that only important aspects will be attended to. One weighting method is to evaluate the informativeness of attributes. The second type of factor is the dependency relationship among attributes. The fact is that a Bayesian model with independence assumptions among attributes (i.e., naive Bayesian) would deteriorate decision performance if some attributes are indeed statistically dependent. The third factor is related to the calculation of the importance index of each stored class. Cases with high ranking values are retained or used for other applications (e.g., model design).

Relationships among attributes and cases may be inductively learned from stored cases. Learning methods differ according to evaluation schemes and search strategies [Langley, 1994]. Evaluation proceeds in two ways: First, learning may be based on statis-

tical information in stored cases used for training. Second, learning could take advantage of cross-validation procedures that repeatedly divided training cases into two parts with one part for training and the other part for evaluation. The two schemes are also referred to as *filter* and *wrapper,* respectively. Due to its complexity, the wrapper evaluation scheme is applied only to learn binary weights. In the current version of NAC, a greedy search strategy is used for most situations. However, in the analysis of dependence relationships, due to the non-monotonic nature of probability computation, exhaustive search may be needed.

5.1 Weighting attributes

The key feature of the proposed approach to attribute weighting is to evaluate the measure of informativeness of each attribute. In the current version, weighting is carried out in the entire feature space, rather than in several sub-regions. Suppression of irrelevant features generally improves performance.

The informativeness is measured in two ways. In the first way, weights are computed with jcp and no cross-validation is involved. An attribute has high discriminability if the jcp of one class is very dissimilar from those of other classes. The similarity metric defined in section 3.1 is used as the measure of discriminability of two classes. The overall discriminability of an attribute j is equal to the summation of the similarity measures for all pair of classes. Finally, the weight associated with a feature can be obtained by homomorphically transforming the discriminability of attribute j from the [0,2] interval to the whole real line.

The second approach is to select the subset of attributes via cross-validation procedure. The selection process is carried out by comparing error rate of tests with different subsets of attributes. Start with the null set. One attribute is added to the attribute set at the end of each test run. Instead of randomly picking up an attribute to test the order of attributes to be selected is ranked according to the measure of informativeness. That is, the more informative attribute is selected earlier. For each run, the training data is randomly divided into two parts. This division is repeated a number of times. The average correct test ratio of divisions is recorded and stands for the utility score of the subset of attributes under consideration. The final selection of the desired subset of attributes is the one with the highest scores. Often, a large percentage of the features can be suppressed.

Our experiments with letter-recognition tests indicate that estimation, based on both kNN and Bayes analysis, using attributes with adjusted weights yields better results than those not using adjusted weights. The improvement of error rate is around 8 percent on average of 10 repetitions with 1000 samples (50 test-percentage used). Binary selection yields better results, but requires extremely longer training time than the weighting method. For some distributions, weighting may be counterproductive. The issue concerning the discrepancy between exhaustive search and the proposed informative search has not yet been analyzed in NAC. Incremental weight adjustment has been extensively discussed in neural network literatures and is not included in the current version of NAC.

The difference between the proposed weighting method and value-difference metric (VDM) [Stanfill & Waltz, 1986] is that the similarity metric of the former computes the difference of distributions between two classes whereas the metric defined in VDM evaluates the difference of two values of an attribute (for data of nominal scale). The proposed metric is similar to the disorder measure of C4.5. The weighting procedure was done by first dividing each class into sub-classes via clustering method and then learning the informative attributes based on samples of sub-classes. This is currently under investigation.

5.2 Analyzing dependency relationship

The objective is to discover a partition of attributes that best represents the stored cases by exploring underlying dependencies among attributes. We consider two aspects. In the first aspect, suppose each case has a category label attached. For nominal-scaled data, the proposed approach derives the dependent relationship between attributes on the basis of improved error rates for test results. The improvement is measured before and after attributes are joined. This approach iteratively joins together different attributes and selects the combination with the highest predicted accuracy. Search halts when the join does not yield better results. Unlike attribute selection, at each test run, a partition, rather than a subset, of attributes is under evaluation. Also unlike attribute selection, no priority is assigned to attributes. Hence, the current problem deals with a much larger search space. In the case of an ordinal or interval scaled measure, Parzen functions are used to estimate probability density for each attribute. Following the same steps, we repeatedly evaluate with different partitions of attributes.

In the second aspect, no information about category label is available. We may employ a statistical test t to test dependency when data is in nominal form. One example is the κ^2 (chi-square) test with a given significance level. In this approach, the joint probability (e.g., *Pr(A,B)*) is assumed to be the expected value and the multiplication of probability density of each constituent (i.e., *Pr(A)Pr(B)*) is the observation. The hypothesis is to test how well the observed values fit the expected values.The search strategy is the same as before. This approach is also applicable to the case when a category label is available. The Kolmogorov-Smirnov test can be used when data is at least ordinal level [Sachs, 1984].

The approach for independency test when a class label is available is discussed in [Pazzani, 1995]. Test results with a congressional voting record data file showed 3 to 4 percent improvement. However, one concern with this approach is that the joined attributes quickly take up huge amounts of memory. In fact, the memory space required for a fully joined probabilistic distribution is exponential proportional to the number of values of attributes. The second problem with the join operation is that it's hard to tell from learned results whether the change in error rate is truly due to a dependency relationship or other factors. The halting condition for search is based on the parsimonious criterion, not on any probabilistic properties. We may construct a network representation [Pearl, 1988] from the learned results for better conceptual illustration of the relationships.

When attributes of an input case are of different types, trait analysis or profile analysis may be needed for dealing with probabilistic dependency. Here, we briefly describe an

approach for handling attributes of different scales. (This approach has not yet been incorporated in the current version of NAC.) We use Bayes rule (i.e., $Pr(A|B) = Pr(A,B)/Pr(B)$) to estimate the conditional probability distribution. If A is of interval and B is of nominal scale, $Pr(A|B=b)$ is a Parzen function described by instances that satisfy $B=b$. On the other hand, with A nominal and B interval scale, for a particular value of $A=a$, the denominator of $Pr(A=a|B=b)$ is described by the Parzen function of $Pr(B)$ with all instances and its numerator is estimated by instances satisfying $A=a$. By the same token, we can estimate probability distributions when attributes of ordinal type are involved.

5.3 Calculating typicality

This method differs because it assigns weights to each stored case or sample. The idea is based on the assumption that data points of high typicality give better prediction. Therefore, a relatively small set of representatives are selected from a distribution and the reduced set hopefully maintains a predictability similar as the original set of cases. Typicality is also used as a criterion for selecting exemplars or representatives.

In DM, typicality of an instance is determined by the ratio of the number of times the instance is referenced (e.g., evaluated as the nearest neighbor to other points) and the number of times it is correctly referenced. With PM, typicality of an instance can be computed in terms of its membership. Suppose the instance is in class Ci, membership is the summation of the ratio between the likelihood of this instance and the maximum possible probability of Ci, i.e,

$$P\langle F_j = j|C_i\rangle / (max(P\langle F_j|C_i\rangle))$$

Although the measure of typicality is computed via a *filter* scheme, it can be obtained using a cross-validation procedure. Typicality in unsupervised learning is calculated by selecting the set of representatives with a distribution similar to the original distribution.

6 Examples of The Use of NAC

Example 1 (classification)

To use a classification method, click on the Data button to load a data file and then specify values for functions in the Experimental Setup. Last, click on Train-and-Test to start running. Test results will be shown in a pop-up window.

Suppose the letter-recognition (LR) test of the CUI data base is selected. The user clicks on the Data button and specifies the number attributes in the input data (i.e., 16 for LR), white space characters used in the data set, and the data type (i.e., numerical for LR). The user then loads the filename (e.g., ll000). The user might click on the Minkowski similarity function since the values in the LR test are of interval scale. For comparison, the user selects a Random Number Seed that is greater than zero (e.g., 3). Using the kNN decision rule with k=1 and under 50% File Test Percentage, the error rate is 66.8%. With

the same experimental setting, the user may test the attribute weighting method by clicking on Slt_Wgt and choosing the option *weight-all*. The user then re-runs the test. The error rate becomes 77.2%. The result of the second test is shown in a different pop-up window. To switch the test data file, use the right most mouse button to click on Data, select Clear Dataset and go to load new data file.

Example 2 (clustering)

The steps for using the clustering methods are almost identical to those for classification. The only difference is that there is no need to set File Test Percentage since all data are used for clustering. After clicking on the Clustering setting, the user is asked to select between a *density* or *probability* model, between *normal*, *manual* or *automatic* mode and between *incremental* or *hierarchical* form.

Suppose that the animal categorization data set [Martin & Billman, 1994] is used. This data set contains information about 10 animals (i.e., macaw, leopard, finch, hamster, pigeon, goldfish, guppy, catfish, gopher and angel fish). Each animal is described by a list of nominal scaled attributes - found, food, covering, legs, mobility, reproduction and appearance. Suppose automatic mode and probability model are chosen. The suggested number of clusters is 3 with the partition {finch, macaw, pigeon}, {angel-fish, goldfish, guppy, catfish} and {hamster, leopard, gopher}. The cluster label of each category is given in the form of an integer. The same results are obtained by using a density model. The probability distribution associated with the current partition is available.

7 Combined Use of Different Functions: On-going Work

The objective is to use classification, clustering methods and other adaptive functions in combination. Among many possible combinations, one is to select a clustering method first and then classify new cases based on results obtained from clustering. One problem is generating new labels. How would those labels be recognized or accepted in different domains? We currently assume that some training samples with externally given labels are available and can be used to adjust and tune the internally generated ones. For instance, the cluster labelled with "24" may be renamed after the class label of the majority of the training samples assigned to the cluster.

The organization of cases with different perspectives needs to use clustering methods as well as methods for analyzing dependency relationships. The fact is that those attributes may reflect various perspectives from which they were extracted or selected. In clustering, using too many dependent attributes may render the result ambiguous. There are two reasons for this. First, the measure of similarity is perspective-sensitive. For instance, Guinea pig is closer to Leopard than Macaw from the animal taxonomic perspective whereas it is closer to Macaw than Leopard if our perspective is pet. Clustering should be carried out with respect to a particular perspective. Second, if the number of attributes with one perspective significantly exceeds that of another one, the result of clustering will likely be dominated by the one with more attributes. Hence, we need to separate attributes and organize cases with respect to different perspectives.

In our approach, the property of perspective is similar to the notion of a factor or a hidden structure, i.e., subsets of closely related attributes become statistically independent once the hidden structure is known. The desired hidden structure corresponds to a partition of cases. We apply clustering methods to the derivation of its property. We assume that attributes that belong to the same perspective are dependent.

The outcome may contain several subsets of attributes. Those subsets may or may not be exclusive. To show the result, we use the same example of animal categorization. After applying dependence analysis, the subset of attributes {found, food} as well as {covering, legs, mobility} are selected. The first subset generates two clusters and the second one generates three clusters. With externally provided information, the first subset can be identified with the notion of "affinity" with two values <pet,wild> and the second one corresponds to "species" with three values <bird, fish, mammal>. The notions of affinity and species corresponds to two different perspectives. From the perspective of species, attributes {covering, legs, mobility} are thematic while other attributes are contextual. In a probabilistic network model, a perspective is represented as a top node. In this example, the top nodes include "affinity", "species", "reproduction" and "appearance". "Covering", "legs" and "mobility" are the child nodes of "species" and "found" and "food" are attached to "affinity". The top nodes may share child nodes. With each perspective, the joint probability density function associated with links connecting the parent and child nodes can be determined. By adjusting the clustering criteria, we could obtain different taxonomical structures.

8 Summary

NAC was designed as an experimental tool that supports basic adaptive retrieval and indexing functions. Important functions of adaptation involve weighting/selecting attributes, learning dependency relationships and calculating typicality for each stored case. Informativeness measures are used to guide search. In dynamic environments, the informativeness of attributes varies. Hence its value is updated in response to environmental changes. For non-parametric estimation, new cases are simply stored, while for parametric density functions, the joint conditional probability density is calculated using a Dirichlet distribution under the conjugate prior assumption. In NAC, the computation of similarity is viewed as evidence combination, rather than simple vector operations. As we just mentioned, two influential factors for evidence combination are the informativeness and dependency relationships of attributes. Case organization utilizes both classification and clustering methods.

For clustering, NAC supports incremental and hierarchical modes as well as kNN and probability methods. An adjustable version of the incremental form was adopted. New indices for cases are generated by using clustering methods. New indices may be either in the form of new terms or in the form of a probability distribution. The case structure can either be a probabilistic network model with new labels as the parent node and attributes as the child nodes or be a discriminant network with one clustering outcome as the parent node and its next generation of clustering outcomes as the child nodes. Case retrieval with new indices is expected to be more effective and coherent. We mentioned that information

about perspective needs to be incorporated into case organization. The perspective information was represented in terms of hidden structure and the results suggest that cases should be organized in multiple ways. As a result, different cases could be retrieved from different perspectives.

NAC facilitates different response modes. The user may choose automatic clustering which requires the least intervention or the trial-and-error (manual) mode where the user is asked to provide information at each step. Many considerations and on-going examples were mentioned in each section. The current version of NAC deals with data represented as a feature list. The immediate goal is to explore processing capabilities for nested or structured representation using more structured objects from domains such as engineering design.

9 References

Anderson, J. & Mates, M. (1992) "Explorations of an Incremental, Bayesian Algorithm for categorization". Machine Learning, Val 9, N0. 4, Oct. 1992, pp 275-308.

Atkeson, C., Moore, A. & Schaal, S. (1995) "Locally weighted learning". (submitted) Artificial Intelligence Review.

Cheeseman, P. & Stutz, J. (1995) "Bayesian Classification (AutoClass): Theory and Results". *Advances in Knowledge Discovery and Data Mining* (eds. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R.), The MIT Press, pp 153--180.

Cooper, G. & Herskovits, E. (1991) "A Bayesian method for the induction of probabilistic networks from data". TR SMI-91-1. University of Pittsburgh.

Croft, W & Turtle, H. (1993) "Retrieval Strategies for Hypertext". Information Processing and Management, 29, pp 313-324.

Del Favero, B. and Fung,R. (1994) "Bayesian Inference with Node Aggregation for Information Retrieval". The Second Text REtrieval Conference (TREC-2), pp 151-161.

Harrison, P., Chang, L. & Meyrowitz, A. (1994) "Memory Organization for Case-Based Reasoning", Proceedings of World Congress on Expert Systems, 1994.

Fisher, D. (1987) "Conceptual Clustering, Learning from Examples, and Inferences". Proceedings of the 4'th International Workshop on machine Learning, pop 38-49. Irvin, CA: Morgan Kaufmann.

Friedman, J. (1994) "Flexible Metric Nearest Neighbor Classification". TR Stanford University.

Fukunaga, K. (1990) "Introduction to Statistical Pattern Recognition". Academic Press, Inc.

Jain, A. & Dubes, R. (1988) Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, NJ.

Jobson, J. (1991) Applied Multivariate Data Analysis. Springer-Verlag.

Kolodner, J. (1993) Case-Based Reasoning. Morgan Kaufmann.

Lagley, P. (1994) "Selection of Relevant Features in Machine Learning". TR 94-3 Institute for the Study of Learning and Expertise.

Martin, J. & Billman, D. (1994) "Acquiring and Combining Overlapping Concepts". Machine Learning, Vol. 16, 1994, pp 121-155.

Parzen, E. (1962) "On estimation of a probability density function and mode", Ann. Math. Statist. Vol 33, pp. 1065-1076, 1962.

Pazzani, M. (1995) "Searching for Attribute Dependencies in Bayesian Classifiers". Proceedings of AI and Statistics Conference, 1995, pp.424-429.

Pearl, J. (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kafumann.

Sachs, L. (1984) Applied Statistics. Springer-Verlag.

Stanfill, C. & Waltz, D. (1986) "Toward Memory-Based Reasoning". Communications of the ACM, 29(12), 1213-1228.

Schank, R. (1982) Dynamic Memory: A theory of learning in computers and people. New York: Cambridge University Press.

APPENDIX 1 (Case button)

In classification, to save the results (e.g., the weight associated with each attribute) of the current test data into a case, click on the option *make* of the Case button before running the test. At the end of the test the user is asked to indicate whether the result of the test is positive or negative. Those cases can be stored in a designated file by clicking on the option *save as*. Because attribute weights are highly domain dependent, applications have to examine whether the index set of test data matches that of stored data as well as ensure that the differences in the distributions of test and stored data are insignificant.

In clustering, if some control parameters are very expensive to compute, those parameters can be stored into a case where the index set is described by geometric measurements about the data set:

CASE i

Index:-

overall_average_distance	: 4.38
overall_variance	: 117.
occupancy	: 160.
average_distance_of_1'st_nbr	: 1.4
variance_of_1'st_nn	: 0.516

Information:-

scale	: 0.01
boundary	: 1.

Constraints:-

pos/neg	: +

To retrieve cases, click on *apply*. Success or failure of retrieval is determined by an interval associated with indices. Values that fall outside the interval are rejected.

APPENDIX 2 (NAC user's information)

NAC Interface Button

[File] This button allows users to access functions to load and save the current experimental configuration so that one may repeat experiments.

[Load Configuration File] loads previously saved configuration file.

[Save Configuration As] saves current experimental settings into a file. Users provide the file name.

[Quit] quits NAC. Options [Default Configuration] and [Save Configuration] are not implemented.

[Data] This button allows users to load and manipulate the dataset that you wish to use for testing.

[Load Dataset] has three functions.

-[Number of Attributes] specifies the number of attributes used for describing an instance in data file.

-[Whitespace Charaters] specifies which characters are to be read as [whitespace]. Whitespace separates attributes. Simply type the characters that you wish to have used as whitespace.

-[Data Type] selects CHAR if values of input are characters

-<Load Dataset> selects input data file.

[Clear Dataset] clears input data.

[Dataset Browser] displays weights of attributes in the textpane.

[Train and Test] This button activates procedures for running the actual experiment. Message regarding the test results will be poped up in Textpane. It includes experimental configuration, number of input data, percentage of test and percentage of correct estimation.

[Detailed Results] shows information of estimation of each testing instance.

[Selected Instances] prints the information about the most "typical" instance selected from each class

[Graph] <not implemented>

[Case] This function includes choices of "add", "save" and "apply". The user selects "add" to record the test result as a case, "save" to create a case file storing cases and "apply" to load the case file and display stored cases if needed. In the current version of NAC, this function is used only to store test results from classifictaion and clustering.

[Reset] This button re-initiates NAC.

[Help] It provides this user's guide.

Experimental Setting

[C/C] choices Users may use NAC for the purpose of classification and clustering.

[Classification] This function is for decision-theoretic based retrieval.

[Clustering] This function is used for re-indexing.

-[Method] Similarity is based on the function of nearest neighbor distance and the size of cluster. Other options can be k-means and single-link method.

-[Mode] It includes Normal, Manual and Auto. Choice depends on the degree of external intervention. For instance, Auto requires only input data file.

-[Increment] The choice is between batch and incremental modes of clustering. If "Combined use" is selected, NAC runs clustering first and then classifying data with respect to the computed clusters.

[Number of Neighbors] This sliding bar sets the value of k for the K-th nearest neighbor algorithm.

[Random Number Seed] This control allows you to set a seed so as to randomly divide input samples into partitions, e.g., testing and training. If it is left as zero the random number seed will be determined by the clock. Partitions will remain unchanged for the same test if the seed that is greater than 0 is chosen.

[File Test Percentage] Use this sliding bar to set the percentage of the data file that you will use as test cases.

[Similairty Functions] Select one of the three options for measuring similarity of different scales of data.

-**[Minkowski]** the measure of difference is for data of interval scale or above.

-**[U-test]** for data of ordinal scale.

-**[Interset]** the intersection or matching operation is used data of nominal scale.

[Normalization Function] Normalization is used to adjust scales of dimensions.

[None]uses original input data.

[Linear]adjusts values of input data to within the range of [0,1].

[Standard Deviation] computes the unbiased variance.

[Decision Function] Decision function calculates final results on the basis of values associated with attributes.

[Direct kNN] employs the standard voting kNN decision model.

[Bayes Statistics] evaluates in terms of Bayes framework. Information necessary to Bayesian reasoning is derived from input data.

[Adaptive Critics] This function is to enhance retrieval accuracy by evaluating importance index.

[None]

[Weighting]

-**[weight-all]** evaluates importance of an attribute based on probability distributions of all classes.

-**[selection]** selects a subset of attributes via cross-validation

[Dependency Test]

-**[CITest]** conducts dependency/conditional dependency test.

-**[CbAttr]** conjoins several different attributes if the join yields better predictive accuracy.

[Typc] calculates importance index of each instance.